Corellium

Video Webinar

Episode 13: *Experimenting with Messaging App Vulnerabilities*

Full Transcript

(00:00)
Brian Robison: Good morning, good afternoon everybody. Thank you for attending today's webinar. Again, we still have a bunch of people joining in, but the good thing is, it's my portion of the webinar first thing. So there's not a lot of great information coming across the wire anyway. So thank you for taking the time out of your day to come and learn about something new that we are very, very excited to release to you today. And I am Brian Robison. I'm Chief Evangelist here at Corellium and I am joined by Chris Williams who's a researcher on our team, but also heads up TFP0 Labs and our big guest speaker today, and we thank him very, very much for his time today is Marco, who's a senior security researcher with the lab, TFP0 Labs, and he is going to be the main one walking us through our topic of today. Good morning gentlemen.

(01:08)
Marco Chomut: Thank you for that great introduction, Brian.

Brian Robison: Alright, so let's go ahead and get going because we want to get to the most interesting thing that's happening today. So let's just go ahead and do a little housekeeping. As always in a webinar, you guys are all muted, but we want engagement with you. Both Chris and Marco can answer questions. We can interrupt during the presentation and address any questions you may have. So please, please utilize that Q&A tool widget on the Zoom screen and please feel free to ask us questions, ask Marco questions while he's presenting. Either Chris or I will be available online answering questions via type or we'll pause the presentation and ask Marco the question for you. So please utilize the Q&A tool for that. The webinar is obviously being recorded. Many of you might've heard that start, but it is being recorded and will be available on demand on our website shortly so that if you miss something or you want to share this content with your team, it will be available as all of our presentations consist of slides as well as live demonstrations.

(02:32)
We're going to do our best as always to make sure that our terminals and our SSH sessions and our text and things like that are fairly clean and legible, but sometimes we just can't change the size. So we'll spend a little bit of time trying to explain what's going on, but again, it's being

recorded in high definition. So if you want to go back and zoom in and look at something at a later time, you're very welcome to do that. And with that I'm going to go ahead and kick into the section that I'm going to talk about today. And it's something that I think is, at least for me, it's very exciting. And for those of you who potentially remember about a year ago we introduced our very first vulnerable application called Corellium Cafe. And Corellium Cafe is really targeted to app pen testers, security researchers, folks like that who are looking for misconfigurations, vulnerabilities, issues with data in motion or data at rest and potentially want to get into a little bit of reverse engineering at the app level.

(03:40)
And it's an intentionally vulnerable application built for iOS and Android and we've been using it for demonstrations, we've been using it for pen test trainings and all that other kind of stuff and it's been available to a lot of you in the past. However, today is when we're going to really launch to the world what we call our solution Guided Scenarios. So we have launched a webpage, it's just Corellium.com/scenarios and it is going to have three different scenarios that are guided for you. So it's going to help you maneuver through how to use Corellium, maybe even third party tools to do some of the challenges and tasks that are in there. And all I'm going to do, I'm just going to pop over and just show you a live demonstration, kind of the site as it exists this morning. So this is the scenarios page and when you land on the scenarios page, there's going to be, like I said, three different scenarios.

(04:48)
The first one is Corellium Cafe. So again, a year ago we built this app and we put this scenario out there, but not a lot of people knew about it yet. We linked to it in some of the new builds of the application, but it's now going live. And essentially what you can do is you can go into this scenario and then you are able to choose a different path if you want to go Android or if you want to go iOS—basically each one of these guides are going to walk you through. Maybe you're new to Corellium and you want to understand how to use the product. So it can be used for logging in and creating your first virtual device. It's got kind of a walkthrough guide feel to it. Or if you're a little bit further along and you're already a Corellium customer, then you can skip down and just get right into downloading the Corellium Cafe IPA and get going on your testing.

(05:51)
And it gives you a couple of ideas of how to use some of the tools and how to find maybe one or two of the vulnerabilities. But there are many more vulnerabilities in the Cafe applications both on iOS and Android that you're going to have to use your own skills to go figure out and find them at the very, very bottom. I will leave you a hint. At the very, very bottom of the page there is an option to check your answers. Maybe if you didn't quite get all of the vulnerabilities or you want to check and see how things are going, you're able to get those answers from us and they're password protected, so you have to reach out to us to grab 'em. But we do provide the answer keys and these are great for skills development, team trainings, all that kind of fun, those fun things. So I apologize, my Chrome browser.

(06:55)
So let's go back and so today we are introducing you to the second release, not of Cafe, but of a new application that is designed for vulnerability researchers and primarily exploit development so that you can basically go in, find a vulnerability and see if you can actually create a real exploitation of that vulnerability to yield some data leakage or whatever you're going to try to do. And that's what we're here to do today is to introduce you to a new vulnerable app called GlitchChat. Now this is really cool because it's really designed around a zero-click type of framework where there is really nothing that an end user is going to have to do for you to, as the attacker, gain access to data on that victim devices. And right now, this is only for iOS at this point, but it does give you all of the things that you need here.

(08:05)
You can download the full package, which includes, because it is a messaging app, there is a server component to it where it allows you to send and receive messages to and from the application that's running on the virtual device. So you can download the full package. It walks you through,setting it up, giving you ideas of where to go, some hints on things to look at. But again, this is going to rely on your skills and your knowledge on vulnerability research and analysis and exploit development to kind of finish up on your own. So we're very excited to release the Scenarios page. So again, if you just hit Corellium.com/scenarios, you're going to land on that page and you're going to be able to go down into any one of those guides that you want to. And I do want to let you know that third one, which is the testing mobile malware, conducting mobile threat research.

(09:14)
That guide is under process right now. I'm working on it as we speak and hopefully that will go live as well very, very soon. So scenario guides. So thank you very much for taking a few minutes of our day and listening to me launch those guides. And with that it is time for me to be quiet and let it switch over to Marco who's going to take us through the rest of the conversation. So again, Chris and I are here for questions in the background. Please use that Q&A widget tool to send us any questions or you can send us private comments there as well. So Marco, I'm going to stop my share and let you take control and off you go.

(10:05)
Marco Chomut: Thank you so much Brian. I appreciate the introduction and I am also very excited to be here today and to talk with you all about GlitchChat and how it can help you out no matter where you are on your journey as a professional, as a student, wherever you are in life or in your professional career. And so let me start sharing my screen over here. And as Brian mentioned, please feel free to ask any questions. We'll do our best to intercept those and to be able to answer them. But otherwise if there's something that's a little pressing or a little necessary, maybe I went over something too fast or someone wasn't able to follow something, please feel free to jump in and then Chris or Brian can interrupt me. It won't ruin the flow of anything. So if something comes up, please bring it up pretty much right away or at the end we'll definitely leave some time to answer any questions that you have.

(11:07)
And so to kind of go over what I'm going to be talking with you today about, we're going to go a little bit into the background for why we wanted to develop another application on top of what is already offered with Cafe. We'll kind of go over what GlitchChat is kind of at a very high, high level how end users might be able to use it, not necessarily you in this training program. We'll go over some of the key vulnerabilities that we built into GlitchChat that you're going to want to be able to learn about and then eventually be able to exploit by the end of the program. Then kind of how the training program will go for you if you choose to start getting involved and complete some of these exercises yourself. And then the more fun part of it, which we'll actually show you a live demo of it in action using Corellium and on the command line and with some scripts that we can have to compliment your journey in exploiting GlitchChat.

(12:12)
And so to go over why we wanted to build another intentionally vulnerable application, we still see a huge shortage of skilled cybersecurity professionals in general, but especially on mobile platforms such as iOS. And conversely, along with that, there's still a huge growing demand for well-trained security researchers. And along with that gap, there's not a lot of modern training resources along the way. There might be an odd blog post here and explanation there, but not a lot of hands-on resources necessarily. And that's really what we wanted to be able to provide for you is this practical hands-on experience that you can use with a real world-ish, close enough to a real world application, that is going to mimic pretty closely to some actual challenges that you're going to face in the wild, whether you're on the offensive side of the house, the defensive side of the house, or you're just curious about security in general.

(13:15)
Now to go over what GlitchChat is, we kind of envisioned this in a fictional universe that is oddly enough in the recent weeks and months starting to come a little bit closer to reality here. So what we imagine GlitchChat existing is for in this universe there's a lot of scammers that are using AI generated images. And so the United States President has issued an executive order that's going to require all messaging platforms that are within the continental United States to be able to start identifying any images sent back and forth that are potentially AI generated. And they're going to do that based on some metadata that's attached to these images and by the tools that are generating these images. And so we imagine that in this fictional universe GlitchChat is kind of the first application that's complying with this executive order, but because it was the first out the door and the first to comply with it, it was maybe a little bit of a rush job.

(14:16)
Maybe they didn't use some of the proper APIs that they were supposed to be using, maybe they overlooked a security vulnerability here or there to be able to get it out the door. But that in turn means that you then have stomping around to be able to explore and to be able to exploit some of these vulnerabilities. So how GlitchChat kind of looks in general, there's not so much glitz and glamor to it, it's pretty just functional. On the left hand side here, you kind of have your

landing page where you're going to be connecting to a server. For those of you that remember like IRC servers back in the day, even though they do still exist and are actively used, that was kind of the more common scenario back then where, OK, I have an actual messaging server that I'm going to be able to connect to with a username.

(15:05)
And then in the middle here, if you've ever used a messaging application, you should be very familiar with this. It's sort of just a list of your active conversations that are going on. You're able to message a new user, go through your contacts. And then finally on the right hand side here we have the drill down detail view of an actual chat conversation occurring with the little bubbles going back and forth, being able to write a new message and attach maybe some images or some files to this message that's going on. Now, as for why GlitchChat exists in this fictional universe, of course users love being able to send photos back and forth with each other. And so this is sort of how a normal attached photo or image would look within a conversation view. But then if the program detects that the image is AI generated—you can ignore the red here.

(15:59)
It's not that it's going to block out the image by any means. In this case, the AI generated image just happened to be a bunch of red pixels over and over again. But the interesting part here is that interspersed message in the middle here, the GlitchChat as an application is going to be able to parse that metadata from the image and say, Hey, I think this image was AI generated. And just leave that as a message to you. You can do with that information however you may. We're kind of just saying, Hey, we're letting you know that it was AI generated. We detected that. Now as for some of the vulnerabilities that made its way into GlitchChat because they wanted to just get it out the door at the last minute, these are then the kind of main scenarios that we see you exploiting as you're making your way through GlitchChat and exploring through it.

(16:50)
In the first case here, after you've ever used a popular messaging application, whether it's iMessage or Signal or WhatsApp, a lot of the time if a link to somewhere is presented, whether it's like a URL to a website, it will try to do a little convenience feature where it'll background fetch that URL, it'll render the page in the background and then it'll show you a little preview of it so that it's good to know if you want to see what you're going to be clicking on. You want to know that it's not taking you anywhere sketchy. So in many respects it's a good thing, but based on how this was built into GlitchChat, it may have not necessarily been done in the right way. And so while they had good intentions when they were building this application in this fictional universe, it's definitely going to be ripe for a little logical exploitation here.

(17:45)
The second scenario that we have built in is what's called a custom URI handler. And for those of you that are not familiar with this, this is used a lot of the time. In this case, I'll show you an example from a popular real estate application called Redfin where if someone sends you a listing to say that's like a home for sale or you happen to be browsing along some social media

and you click on it, if you already have the app installed, it's going to automatically redirect you to the application that you have installed on your phone and really drill down and deep dive into the page that they have within their application for this listing. And so this is a convenience feature that allows them to say, Hey, I noticed the user was clicking on a link. My app can handle that link, so let me do that instead of just showing you a normal webpage.

(18:38)
And again, this is a convenience feature, it's really nice for the end user, but as you'll come to see within GlitchChat, it may have not been handling the URL parsing in the right way or it may have been exposing things that it shouldn't otherwise have been available to an end user that shouldn't have access to it. And then the final vulnerability here really has to do with the entire purpose of GlitchChat's existence, which is its image rendering, its image parsing library. When you start drilling down into it, you're going to see that it's using a custom fork of a very popular open source library called the PNG and that it is using this to be able to parse out some of the metadata, try to find if it's an AI generated image, but because this is happening at more of a lower level, it's like a c library, it's going to be vulnerable to some of your more classic exploitation techniques.

(19:37)
As for the training program itself, as Brian had mentioned earlier, please feel free to go into this scenario guide. You're going to be able to start drilling down into this and reading through this yourself, but to give you an overview of how it will go for you if you choose to do so, it's going to get you very familiar with Corellium as a platform, be able to guide you through actually setting up a device and getting GlitchChat installed. You'll be able to play around with GlitchChat yourself as if you were an end user, maybe start sending some messages and some images back and forth and then get you introduced to the idea of, oh, in general, how do I start reverse engineering an iOS application using some popular open source tools. Then just as we had gone over right now, it's going to take you through each of these major vulnerabilities.

(20:26)
In turn, it's going to take you through that URL link preview, get you introduced to some Python scripting to be able to automate how you're exploiting these vulnerabilities. Then taking you to that custom URI handler, it's going to introduce you to the idea of Frida Hooks and being be able to dynamically introspect an application as it's running, change things around while you're hooking things, and to be able to really introspect and to be able to craft a vulnerability that you're able to then exploit using. Again, another automated Python script. This is also unlike the other two vulnerabilities. This isn't necessarily a zero-click. It's going to involve a little bit of a social engineering aspect as well. So if you're not familiar with that, it'll get you used to the idea of, oh, maybe I have to entice the user to click on something that they wouldn't have otherwise done.

(21:21)

And then finally, as I mentioned earlier, that custom image vulnerability using the fork of the PNG, this is going to really take you through, while some of the previous vulnerabilities were more maybe high-level logic bugs, this is really going to give you a good deep dive into some of your classic exploitation techniques. You're going to be doing a buffer overflow here. You're going to be identifying that ASLR is active on the system. And so to be able to defeat that, you're going to need to be able to look into some information leaks to be able to make sure that you are aligning your addresses correctly. And then finally, because of how lockdown of the platform iOS is and not allowing code execution, most of the time you're going to be building up a good first step of a ROP chain here to be able to actually do the final exploit and to be able to get whatever data you want to be able to offer the device.

(22:21)
And then finally, we'll leave you off with any resources that we think might be useful for you in the future. Kind of connecting these back to, hey, sure, these were maybe some contrived scenarios within a more traditional messaging application, but these sorts of attacks are done in the wild, they were done in the wild, they are continuing to be done in the wild, and it's a great way to bridge what you've been doing in this controlled scenario to what's actually happening in the real world. Now let's go over to the exciting part of today where we'll show you the actual live demo of GlitchChat in action. And so to start us off here, let's go over to Corellium and based on how familiar you may or may not be with Corellium, I've already taken the liberty of setting up this demo device. It takes a little bit of time, so I didn't want to spend a lot of today going over that.

(23:18)
But once you do go through some of those initial setup steps that Brian was showing you earlier in the Corellium scenarios guides, we have an exemplar iPhone 7 device over here. It's running iOS 15.8 and by default, a lot of Corellium devices come pre-jailbroken, which kind of allows you to get a lot more developer access introspection. It really lets you get access to a lot more of the internals to really play around a lot more with the device. And I've already taken the liberty of setting up the device itself here on Corellium with setting up my VPN connection to it so that I can demo this all to you today. But as you see here, they have plenty of instructions if you want to be able to set that up yourself or to be able to even SSH into the device. And so to kind of start us off, other than being jailbroken, it is a pretty standard stock install of iOS.

(24:13)
And so what we're going to initially need to be able to do is to install GlitchChat itself onto the system. So we can go over here to the apps panel and then just install the IPA that we're going to provide you as part of the training program. And then if you were to just click on chat itself and launch the application, you see kind of that initial connection screen that I had shown you earlier in the slides as Brian had mentioned briefly earlier, along with this IPA and the documentation. We're also going to be giving you the server component of that. We're not expecting you to be able to have to go to some third party chat server where there's a lot of other people that are training at the same time. So along with the IPA, we're going to give you

this little Python script here that I'm just going to run in the background that's going to be sitting there.

(25:05)
It's going to be taking the messages that one user might be sending to another and it's going to relay them along correctly. But of course, if you're interested, I would definitely encourage you to explore the server component of it, explore the protocol that it's being used. It's a little outside of the scope of why we built GlitchChat, but there's certainly plenty of avenues there to be able to say, oh, could I spoof a message from another user, for example? But like I had said, that's a little out of scope for here today, just something that if you have some spare time, absolutely, please feel free. Go dig in there, see if you can find some vulnerabilities that we didn't even intend to necessarily build into the program. And so here we're going to, I already have my IP address and the port of the server that I set up, and we're going to connect here as Alice to it.

(25:56)
And then we don't already have any active conversations. We just launched this application for the first time, and so we need someone else to talk to. And so while I could have a physical device here with me or set up another Corellium device, what I've taken the liberty of doing is just having a little command line chat equivalent of the UI of the GlitchChat application. And so here I'm just running this little program which is going to say, Hey, I'm going to be Bob and I'm going to be talking to Alice. And so if I were to type, Hi Alice, welcome to GlitchChat.

(26:39)
And we go back over to Corellium and we see that, oh yes, it recognized immediately. I got a new message in from Bob as Alice, and then I can see that I got that message and then I can write Bob back. Thanks Bob. And we see back on the command line, oh yeah, we got that message back from Alice. And so that just kind of confirms that we've gotten everything set up correctly. Our VPN is working properly on both ends, everyone's talking with each other. So to then kind of show you how we might start exploring GlitchChat and taking a look at, we're going to walk you through a little bit today of that first vulnerability, the preview of URLs that are being sent back and forth. And we're not going to take you the entire way through and spoil that, but we'll walk you most of the way through.

(27:33)
It's just so you can kind of see how you might use Corellium and start exploring GlitchChat in general. And so if we wanted to go in here, the first thing that you might want to start doing is to use Corellium's Network Monitor to start intercepting some of the traffic that's happening so that you can start exploring that and see what's going on here. So in this scenario, I've set up the process name to be GlitchChat. I know it's going to be talking on port 1883 to the server that we had set up earlier. And then I want to then—bear with me one moment here. Let me clear out anything that I had previously and then we'll start monitoring and then we can send from Alice's side. Hey Bob, do you have any photos to share from your vacation?

(28:41)
And then as it's capturing this along in the background, what I'm going to do is swap on over to that little Python script that I had alluded to earlier. It's this little chat client if you wanted to run this alongside it, just doing some basic setup of the messaging server that you're going to be talking to. And then you can kind of see here how it's going to be. Don't worry too much about the details here, but it's going to be doing some of the heavy lifting of actually sending the messages back and forth and parsing a message. And so what I've written up is a little version of that script to be able to send an image over from the command line. And so let me run that real fast so that you can kind of see how an image shows up in here, but also what it will look like when we're capturing it through the Network Monitor.

(29:47)
And so if we go back to Corellium over here, we see that the results from the script were that, oh, I was able to attach an image, I was able to send it on over to Alice, and then I believe the Network Monitor should have captured this all by now. So if we stop monitoring and then download, it's going to give us a handy little PCAP file that I'm going to open up in Wireshark here. And then let's start exploring what these messages start looking like when they're going back and forth. Initially, we see that Wireshark is identifying the protocol that it thinks it's doing. And so I'm going to follow the TCP stream here so that—please bear with me if the text is a little bit too small on your screen—I'm just going to be describing what you see here. We might not know the actual binary format of what's going on, but we do see that there's some play text being passed back and forth.

(30:45)
Initially we're seeing that there is something that's being sent to Bob as what looks like a private message. And then we see, oh yeah, this looks like this might be a little JSON formatted plain text with the message itself that we got sent from. And we see, oh, it's saying who the message was from and that, oh, maybe there are going to be different types of messages. In this case, it's a text for the message that Alice had sent to Bob. And then from that script that I had written earlier for Bob to be able to send back the photo to Alice, we see again, there's a little JSON message of Bob saying, Hey, yep, this is an example message from me to Alice's private chat channel. But then we see from Bob, we got a message here and it's just a blob of text that we could probably start exploring here if we wanted to maybe see, oh, is this maybe like a Base64 encoded image?

(31:42)
But if we scroll right on past it, we can see that afterwards it gives a file name of what the file was that was sent over and that the type for this message is an image. And then, oh yeah, I see a little message here that wasn't necessarily in the chat thread, but it looks like from Alice, we get this type of a message called an acknowledgement of the image. And within the message text here we see that, oh, it's giving us maybe a debug message here. I don't know if that was intentional on the part of these fictional GlitchChat developers, but it's certainly nice that they gave us this little thing here that tells us exactly where that image got saved off to on the device.

So again, being able to explore how the protocol works, being able to network capture with Corellium can really give you an idea of starting to dive into some of the internals here.

(32:41)
And just to show you that, oh, this message didn't actually appear here, it is nowhere in the thread. This is sort of an internal back and forth message that's happening from the application to Bob's chat clients to acknowledge that, oh, I received the image, but maybe it shouldn't be necessarily exposing the full file path of how it's saved off that image onto the device itself. And so now let's see what happens if Bob were to send over a URL in this scenario. So if he says, Alice, I think you should check out github.com. And then we go back over to our actual GlitchChat application. We see that I had shown off in the slides earlier. It shows the message itself and it does have a clickable link for Alice to click on to go to GitHub, but then it also shows a little preview window of what GitHub would look like if she were to open it up.

(33:48)
And so that then tells me that it's working. I can see that it's taking a URL, it's identifying a URL within the text message, and then it's going to parse it and try to render it. And so I am definitely curious about the two sides of things here. I see that I'm able to attach images, it saves it off, and I see that URLs are getting rendered here. Now I saw earlier in the sort of Wireshark dump that I've got a full path to where the file got saved off here. So one thing that you might start thinking about here is, oh, well, if it's parsing URLs, could it start parsing file URLs like in an internal URL, not something that's like an external website that's being clicked, but maybe GlitchChat might start rendering something that's like an internal file on the device.

(34:48)
So what I'm going to do is as a entrepreneuring security researcher here, I already have a program a little stripped here that's going to send off that beach vacation photo from earlier. But instead of a real image, let me send over maybe like an HTML page. I'm going to make it pretty simple here. And just to kind of show that I am landing this page correctly, I'm going to make the background be entirely green. And then I'm just going to run a little JavaScript on here and then say, Hey, let me ping an address of mine. So I know that I am running on, I believe this is my IP address right now on the VPN. Lemme confirm that. 10.11.3.2, yes, that's correct. And let me also run a little Netcat server over here. That could be on the receiving end of this if it does render the JavaScript correctly. So in our sending of an image example, instead of this beach vacation photo, let's send over that little test HTML page and see what happens.

(36:05)
So we're going to save this off and we're going to go back to the command line and we are going to—actually, lemme do the Netcat from a different window. We're going to send that message on over again from Bob to Alice, but instead of the actual image, what we'll see is that, oh, it didn't render the image. And that makes sense. It had rendered the beach vacation image earlier, but in this case it was an HTML and the image render probably doesn't know how to handle that necessarily. And so it's just showing this generic gray, it's not the green that we

expected, but what we do know is that if we send a properly formatted link to something that it should then try to render it as a HTML page instead of an image. So I'm going to have Bob then say, okay, I'm going to go over to the Wireshark that we have over here, a capture of where it had saved off the beach vacation image earlier, and I'm going to have him send a message to Alice where it will, oh, this is not going to work correctly in that terminal.

(37:23)
So instead, ah, yes—I have it right over here conveniently. I'm going to, instead of a normal URL such as http://google.com, I'm going to do a file://URL. And this has that full path to that debug message that got sent back to us of the HTML file that we sent over. We told it it was an image, it didn't render it properly as an image, but let's see if I send this, what happens on the GlitchChat. So I send over that URL, of course Alice always could click on it, but it's now doing that little preview image and that's that green that we had made the background color of the page. And so now, okay, it is rendering, it's rendering that page, it's able to parse an internal URL, maybe it shouldn't necessarily be doing that. And then let me go on over to that Netcat listener that I had running and we see, oh yes, we got a hit from it and it was from an iPhone.

(38:38)
And so we're seeing that OK, not only can I send over not just an external URL but an internal URL to a file, but when it renders that page, it's going to be running any JavaScript that's on there. And so I hope you've been following along the way here and we're not going to show you the very final step here, but what we hope you will be able to do, especially with the scenario guide, is to then take it to the final step of the way here, which is to then use that JavaScript and start running it to access other local files that are on the system and then send them off to a nefarious server that's sitting somewhere.

(39:24)
So that's kind of it for how the URL renderer vulnerability will get exploited. And then just to kind of show off, we're not going to be doing sort of this more in-depth walkthrough for the other two big vulnerabilities. We want you to be able to dive in yourself and to be able to explore on your own and not spoil too much along the way, but just to kind of show you how this might look at the end of the day, once you do have a full working exploit, I am going to just run some kind of solution scripts here so that you can get a good idea of what it will look like at the end. And so in this case, let me make sure I have whatever server components that I need running, but then we're going to be starting a new command line chat conversation.

(40:18)
In this case it's going to be a new user called Mallory. And in that second scenario, this is that URI handler. So instead of the rendering within the chat conversation of a link that someone sends to someone else, instead this is the application itself saying, Hey, I handle links that look like this and I can send you to other pages in my application. And so what we'll see here when I fire off this exploit script is that Alice has a new chat thread from Mallory over here, sort of similar to how we were seeing with Bob, Mallory's sending over to Alice a photo from her

vacation, and then she's sending her a link to say, Hey, I want you to look at my whole album. In this case we see that, oh no, it wasn't rendering that link in line like it was doing with other things.

(41:13)
But instead what's going to happen is that if Mallory is able to entice Alice to click on this, it's not going to take her to a full album of anything. Instead Alice is going to get this weird alert message and then we're going to try to do some social engineering here of getting Alice to, if you read this message, so what's going to happen when you're exploring this with Frida and trying to exploit things here is that we're going to be overriding a file with another file. And then so if Alice is persuaded to send back this same photo over here and say, sure thing Mallory, it's not an image anymore. Instead it's this file that got overwritten when Alice incidentally clicked on this link that maybe she shouldn't have. And so if we were to go over to the exploit side of the house, we had this server running here on the side that's going to be intercepting that message that Alice had sent back to Mallory.

(42:25)
And so if I look over to my file system and see, oh, this was a file that I had sent over, Mallory had initially sent over this beach vacation photo and it's only 120 kilobytes, but when Alice sent it back to Mallory, it's 508 kilobytes. And that's because Mallory had sent an exploit over to be able to, instead of it being the actual image itself, it ends up being a SQLite database that's on the phone for all of GlitchChat. And then again, this might be a little hard to read, but just to give you a general idea, it's the full SQLite database that's underlying GlitchChat. And so now Mallory, just from a little social engineering and a little exploit knowhow is able to see messages from say Alice to Bob that she obviously as Mallory should not be able to have access to. And then whatever else you could imagine being in the SQLite database, whether it's sensitive information of Alice or other contacts that she has.

(43:33)
And then let us now move on to the final one, the final exploit here, which is going to be the AI image detection using that libpng fork. And so just like before, this is going to be a script that's going to be sending from Mallory to Alice. And then how it's going to look once you put this all together is that we put in some debug messages here, but it is going to be doing everything that I had talked about earlier in the slide deck, which is it's going to be when you're figuring out and going along with this training program, you're going to figure out, oh, can I identify maybe some ASLR info leaks? Can I put this together to be able to generate a wrap chain to call a function that I shouldn't otherwise be able to have access to? And what this leads to at the end of the day is that this doesn't require any social engineering or interaction.

(44:31)
We kind of see some of those interspersed messages, just Alice seeing that, oh yeah, these images were AI generated, I didn't even click on anything here. All I'm seeing is that nice warning that GlitchChat is presenting me. But from Mallory's perspective, she was able to send

over just with an image attachment to do a full kind of end-to-end zero-click exploit that ended up leaking the entire passwords file from the iOS device. And so that's kind of your end if you imagine this as a sort of capture-the-flag exercise, that's what we see as the final flag. The final success case for you getting through this program is, oh yeah, I'm able to link this file to obviously me sitting over here as Mallory and I should absolutely not have access to what's on Alice's device. So that is an overview of the three major pieces of GlitchChat and that concludes the live demo portion of this.

(45:35)
And so I hope you were able to follow along and I really, really encourage you to go check out the scenario guide that Brian had given you a link to earlier and to start exploring, explore Corellium yourself, get familiar with it, download the IPA of the GlitchChat application, start exploring it, do some static analysis, start running it in Corellium, and then start being able to successfully do some of these exploits on your own of these vulnerabilities that were built into it. And Brian, now I will throw it back to you or Chris if we have any pressing concerns. But otherwise, again, please feel free to post any questions that you have in the chat. We're going to be here for the next little bit. And thank you again for your time today.

(46:32)
Brian Robison: Thank you very much, Marco. That was awesome. As all of you can see, the GlitchChat scenario is very in depth, very well thought out and is going to be a lot of fun to play with, especially as we continue to develop it and put other things in it. But definitely in combination with Corellium will give you something to do when you're getting started with Corellium or maybe you're an advanced user and you just want to sharpen your skills or learn new skills. This is absolutely an awesome tool to use for that. So again, Corellium.com/scenarios is where you gain access to the different scenarios. And then we do want to leave open time for questions. There were quite a few questions coming in. Chris did a very good job addressing most of them. However, while we are addressing those questions and saying our thank yous, I do want to do a couple of things.

(47:34)
One, I'd like to give a shout out to Umit who's on the line today actually with us. He's from Mobile Hacking Lab and they have all kinds of courses and labs over there that do similar things. And perhaps he's even going to begin working GlitchChat into one of the courses that he'll be working on over at Mobile Hacking Labs. So please check these guys out. You can go to them directly or you can get to them from our Corellium.com website. In the training classes, you can see there's several different partners there, including Umit’s or their Mobile Hacking Labs. The other thing I would like to mention, I think we've actually had a couple of questions pop in. Do we plan on making a similar app for Android? Chris, do you want to take that potentially?

(48:27)

Chris Williams: I think the best we can give on that at the moment is possibly. So it's very good to have that feedback because we can bring that back and discuss putting that on our roadmap.

(48:37)
Brian Robison: But at least for right now, this one is iOS only. But again, these things change over time and become more available in different areas. Alright, so the last little thing I want to do is I do want to talk a little bit about next month's webinar. So there was little something that some of you saw in Marco's demonstration. There's a new tool becoming available in Corellium. It was in the bottom left hand corner and it's basically a mobile application security test framework. It's a static and dynamic application pen testing scanning tool. We're going to go into that in depth in next month's webinar and introduce it to the world and it should be available for customers very shortly thereafter. But really we are working on building a tool, basically a framework, that will help application pen testers do what I call automating the mundane.

(49:50)
There's a lot of evidence gathering static analysis checks, even dynamic analysis checks with tools like Network Monitor that Marco showed, pulling those PCAPs in, scouring through them for evidence, things like that that could actually be done in an automated and much easier fashion for you. And allowing pen testers to focus more in the discovering of vulnerabilities and developing exploits and things like that more in the art of pen testing. So we see it as a great opportunity for pen testers to be able to repeatedly and very, very quickly gather some of that core baseline evidence from applications and then move into the more intricate elements of pen testing. So let's see. I think there are any further questions. Please feel free to get them on the screen if you wish. We do have a few more moments, but we do thank all of you very, very much for coming and hanging out with us today.

(50:56)
We hope that the content today that we shared, while we are very excited, hopefully it also excites a lot of you out there in the community and I don't see anything really popping in anymore. So again, I just wanted to say a final thank you to all of you. Thank you to Marco and Chris for coming on and giving a wonderful demonstration, a great talk. And then keep an eye out for next month's webinar invite where we're going to be deep diving into automated security testing using Corellium's new capability. Until then, thank you for attending and we will see you very next time. Happy to see you and keep testing and keep working and keep coding and keep breaking things. Thank you very much. Have a great day.

##